THE UNIVERSITY OF ALABAMA IN HUNTSVILLE

NASA/MSFC GRANT NCC8-016

Annual Report

KNOWLEDGE-BASED MACHINE VISION SYSTEMS FOR SPACE STATION AUTOMATION

August 1, 1988 - July 31, 1989

Dr. Heggere S. Ranganath
Principal Investigator

Laure J. Chipman

Department of Computer Science
The University of Alabama in Huntsville
Huntsville, AL 35899

August 23, 1989

I Introduction

a. Vision Systems

Vision systems which can perceive environment through sensors and respond with appropriate action or decision have numerous industrial, military and space applications. To place vision systems in context, we must understand the image-to-decision paradigm, the basis of vision systems. In this paradigm, the raw, sensed image, is progressively transformed into an explicit symbolic description of the scene's content with respect to some semantic model which is used in the decision making process. Various stages of image-to-decision paradigm are shown in Figure 1. In stage 1, the system computes local and global image properties such as gradient, texture, histogram, etc. which are useful for segmenting the image in stage 2. Segmentation is the process of partitioning the given image into meaningful regions, surfaces and objects. In stage 3, significant attributes (features) of each region or object are extracted. These features can be used to identify individual scene components. Relational descriptors of stage 4 specify relation of each scene component with others which is very useful, in stage 5, for making decisions.

b. NASA applications for vision systems

Vision systems can be used to automate routine space station operations, thereby relieving crewmen of repetitive tasks. This increases the crew time available for operations which require human

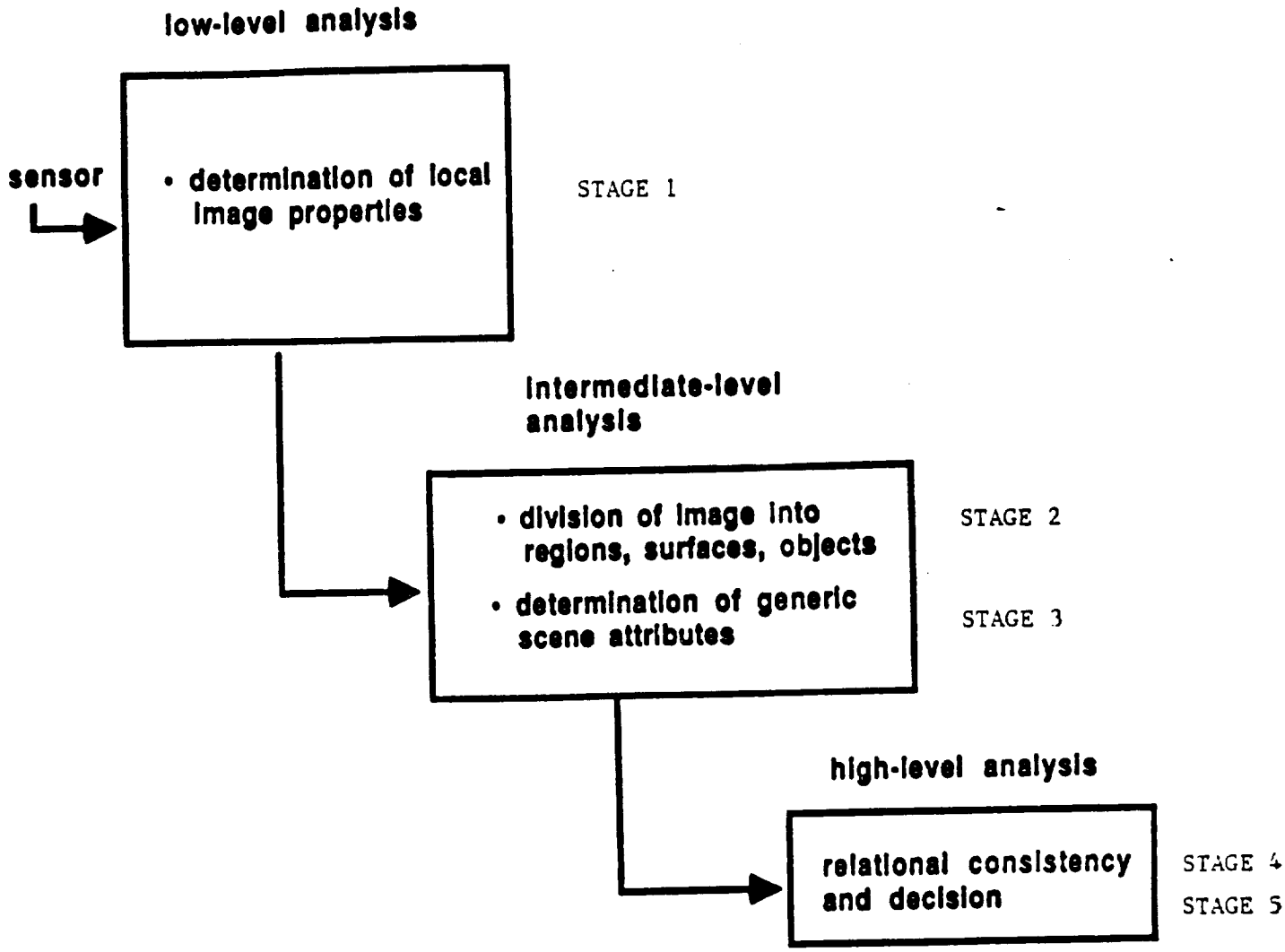

FIGURE 1. **Signal-To-Symbols Paradigm for Computer Vison**

skills. Vision systems can be used to observe experiments, record data and alert astronauts only when necessary. Within the space modules and nodes vision intelligent robots can be used for operations such as locate, fetch, store and repair.

Vision systems are also useful for orbital docking, servicing, assembly and other advanced space station operations. NASA inhouse research shows that providing computer vision capability for the orbital maneuvering vehicle (OMV) offers several advantages: provides independence from docking aids and communication links; eliminates communication time delay for vehicle control and reduces operator training cost for remote control [1].

NASA is also investigating the possibility of using vision system for weather prediction by tracking cloud motion. In short, reliable vision systems are needed to automate space station and other advanced NASA operations.

SCOPE OF WORK

The purpose of this study was to evaluate and provide a realistic assesment of computer vision techniques which have the potential for use on space station and related applications. Specifically, the task included the following:

1. Identification of performance requirements.
2. Evaluation of recent signal processing chips.
3. Selection of candidate systems for further investigation.

   The three major approaches selected for study are:

   a) Conventional algorithmic approach.

   b) Expert vision system approach.

c) Neural network approach.

4. The winning methods in 3 will be simulated to study and verify performance capabilities.

Knowledge Based Approach for Vision

Because of the limited competence of conventional vision systems, knowledge based approach is selected for detailed investigation. Although a basic understanding of vision is emerging, computer vision systems are not as competent in "seeing" as humans. One reason for the limited competence of computer vision systems is the failure to integrate knowledge about the viewing environment into the vision process and the reliance on "weak" problem-solving methods. "Weak" methods, general purpose and domain-independent techniques, were advocated in the earlier days of artificial intelligence; however, these methods did not produce systems which performed as competently as humans. The success of these methods greatly depends on the way the problem-solving process uses domain-specific knowledge to constrain the solution. One of the major results to emerge during the last two decades of AI research has been the recognition of the limited effectiveness of these weak methods to solve hard problems. On the other hand, domain-dependent problem-solving techniques use as much domain-specific knowledge, including procedures, as is necessary to attain the desired level of competence but only within a highly restricted class of problems. The emphasis on domain-dependent techniques evolved into the expert system approach to problem-solving.

Neural Network approach appears to have great potential. However,

this technology is in early stages of development and not yet ready for large scale implementation. A hybrid system which integrates good features of Knowledge Based and Neural Network approaches appears feasible in near future.

The first year research effort concentrated on the development of Knowledge Based Vision System (Expert Vision System). A general description of Expert Vision System is presented in chapter II. A description of the system that is being simulated on perceptics is given in chapter III. Conclusions and recommendations are made in chapter IV.

II. EXPERT VISION SYSTEM

Expert systems illustrate the adage "knowledge is power". These systems have attained a high level of competence in solving a class of problems simply by possessing and using a great deal of domain dependent knowledge, represented explicitly and applied flexibly. Standard expert system architectures, called "shells", have evolved which contain everything needed to solve the problem except expert's knowledge. The expert system builder fills up the shell with the rules and facts used by the expert and the shell provides the inference engine and user interface, explanation subsystem, etc. In short, it is the knowledge which imparts the shell the problem solving power of human experts. The advent of expert shell has greatly simplified the construction of expert systems.

Expert systems exist in diverse fields, from agriculture to space exploration. One reason for their wide applicability is that expert system technology is able to solve many different problems uniformly. The expert knowledge is represented uniformly and a generic inference engine mechanically applies the expert knowledge to the given problem instance, in doing so, derives the problem solution. As long as the knowledge can be represented within the knowledge representation language, the specific domain which generates the knowledge is irrelevant. It seems reasonable to expect that the expert system technology will expand the problem solving capability of computers to new

areas and domains.

Computer vision provides a challenging domain for the application of expert system technology. Very little work has been done towards the development of goal directed, knowledge-driven systems[2],[7]. In order to achieve its goal the vision system must use knowledge about the scene or environment and objects in them, as well as knowledge about the imaging system. Most existing vision systems use no or very little scene knowledge. Also, whatever little knowledge is used is implicitly embedded in image analysis procedures. If scene knowledge and the rules for how it is used is represented explicitly, vision systems become flexible. In this paper, such vision systems are called Expert Vision Systems. An expert vision system uses knowledge to achieve high performance level at every stage of image-to-decision paradigm. In order to get a feel for expert system, consider the following debatable assertions made by Rosenfield[10].

Some debatable assertions: Rosenfield, in his position paper, has made some debatable assertions to stimulate discussion on issues of expert vision systems. An analysis of these assertions highlights the issues and problems associated with expert vision systems and suggests possible ways for designing them. In this section, author states each assertion and then gives his position. The major issues of expert vision systems discussed are : hardware architecture, representation and the use of spatial knowledge.

ASSERTION 1: Expert Vision Systems Are Inefficient.

There is no evidence which suggests that expert vision systems cost more and require more computation. However, we believe they can be more reliable and flexible. Eventually, system performance becomes more important than cost. Low cost systems that do not work are not at all useful.

ASSERTION 2: Any Standard "Expert System" Package Can Be Used To Represent Scene Knowledge; There Is Nothing Special About Spatial Knowledge.

Standard expert systems function in highly restricted arenas in which the domain is symbolic and expertise is well-defined. But much of the vision is pictorial. We must extract useful information and make it explicit. It is difficult to describe shape constraints on natural objects and the spatial relations that hold among them. None of the existing techniques can represent scene knowledge adequately. In short, a scheme to represent scene or spatial knowledge must be developed.

ASSERTION 3: Scene knowledge must be expressed in terms of image properties rather than scene properties, to make it easier to use in analysing the image.

Deriving image properties from scene parts is relatively easy (perspective projection), but, the inverse process is not easy. We feel that scene knowledge must be represented at different levels of detail. Properties such as connectivity and inclusion

which are invariant to the position of the view point must be explored. These properties are common to image as well as scene parts. Topological features seem to be very useful in describing scenes.

ASSERTION 4: If we are dealing with only one type of scene, the scene knowledge required is not great.

Spatial knowledge associated with vision systems which operate within limited man made environments may be relatively small. However, the type and amount of knowledge required may change from application to application. Therefore, knowledge representation shell must be general enough to accomodate wide range of decision applications.

ASSERTION 5: Scene knowledge cannot be used in processing a raw image; the image must be segmented and converted into symbolic form.

This statement is not true. Scene knowledge is useful at each stage of image analysis. For example, knowledge of scene content may be used to select the most suitable segmentation algorithm.

ASSERTION 6: The system will first segment the image, then label the parts, then check the labels for consistency.

We must use scene knowledge to improve the labeling process. When in doubt, we must assign multiple labels to an image part and then seek additional evidence by examining relational

consistency as defined by the scene model. Thus, we can defer our decision until certainty measure becomes very high.

ASSERTION 7: The system will apply a standard set of processes in a standard order to any input image.

The system must determine processes to be applied and the order of application based on data and the goal. The image need not be processed from left-to-right and top-to-bottom. For example, why should the system search for tanks on a lake? Scene knowledge must be used to reduce computation time.

ASSERTION 8: At the segmentation stage, when the system deals with the image, it has to do a lot of computation, and parrellelism is desirable. Afterwards, the system deals only with symbolic data, which requires much less computation.
This is true. The system architecture must provide parrellelism for low-level image operations. We suggest modular design and blackboard architecture.

ASSERTION 9: It may be difficult to formulate constraints that define a given class of objects or scenes, but we can build a system that learns these constraints from examples, just as people do.
At this point in time vision systems that can learn from examples are not feasible.

FEASIBILITY

During the last decade researchers in the area of image processing concentrated on low and intermediate aspects of vision[10]. Last decade's research has produced: methods and algorithms for image enhancement, restoration, segmentation and registration; methods for describing image components and relations that exist among them; techniques for handling time varying images; architectures which can perform several hundred million operations per second(systolic arrays, cellular logic arrays, cyto computer, MPP, ZMOB, etc.); preliminary designs of realtime systems(image correlators, target recognition systems, etc.)

The cost of vision systems is going down. Many semiconductor manufacturers are now producing fast digital signal processing chips at moderate cost. Architectures of these chips are optimized for image processing. For example, Zoran Corporation has announced a chip which can con olve or correlate 256 * 256 8-bit image with a 12 * 12 operator in less than 10 milliseconds. The same chip can compute 1024 point complex discrete Fourier transform in less than 2.4 milliseconds. Several chips can be cascaded to handle larger images and operators. Ten years ago real time image correlation was not this simple. More powerful chips are expected to become available soon. As a result of these chips, design time and cost of vision systems will go down considerably.

In short, the majority of the low and intermediate level

vision problems have been solved. High speed image processing chips are now available at moderate cost. As a result of these achievements development of expert vision systems has become feasible.

MAJOR PROBLEMS

Three major technical problems in the area of expert vision systems are given below:

ARCHITECTURE: It is clear from image-to-decision paradigm that vision systems transform pictorial information in an image into explicit symbolic description of scene components. Early stages of processing is computation intensive and parallelism is essential. If modular design is used processors can be added or deleted as needed. Each processor must be capable of broadcasting information to the rest of the system. A blackboard architecture seems appropriate for vision systems.

KNOWLEDGE REPRESENTATION: Vision systems require different types of knowledge at different stages of image-to-decision paradigm. Much of vision knowledge is pictorial or iconic. It is not easy to represent constraints on natural objects and the spatial relations that hold among them. Spatial knowledge must be represented at different levels of detail. Properties such as connectivity, inclusion,etc. which are invariant to the position of the view point must be explored. In short, spatial knowledge representation theory is needed. Dr. Ranganath already has a

preliminary design for knowledge representation shell.

SPATIAL REASONING: Spatial reasoning is the process of inferring scene information from images which is not explicitly available in the image. Spatial matching is quite different from symbolic matching. Vision systems may infer based on procedures rather than rules. Procedures handle spatial matching better than rules. A theory of spatial reasoning is also needed[10].

III. Research and Implementation for Object Recognition

Much of the work on this contract has consisted of research into machine vision problems and development of a demonstration system. This system implements some of the capabilities that would be necessary in a machine vision system for the robot arm of the laboratory module in the space station.

Much of the early work on the demonstration system consisted of experimentation with methods of achieving some of the lower-level tasks needed. The knowledge gained, about what methods are appropriate and when, can now be incorporated into the demonstration system.

The general research focused for the most part on the problem of scene matching. A graph theoretic approach was found to have merit for a wide range of machine vision problems. The principles of this approach are presented in this report.

This report describes our research and the progress on design and implementation of the demonstration system. Section a is an overview of the problem of object recognition, and the expert systems approach to the problem. Section b provides details of a general strategy for object recognition. Included are sections on the coarse discrimination among different scenes, scene

knowledge representation, and scene matching. Section c describes the design and components of the demonstration machine vision system under development.

a. The Object Recognition Problem

A necessary component of a machine vision system is the ability to make sense of an image, by interpreting its parts as being various objects. A first step towards this goal is provided by the numerous image segmentation algorithms which have been tried over the years. An image can be segmented into parts based on similarities within the parts, of color, intensity, texture, etc., and based on differences among the parts.

Because of noise, differences in camera position, and limitations of the algorithms used, a machine vision system needs to make use of as much knowledge of the expected scene as possible in order to make sense of an image. An expert systems approach to object recognition makes use of a knowledge base of known objects to search for objects "intelligently." An example of a search which does not make use of "intelligence" is a template matching approach where a scene is convolved with a template of the object to be located, and the area of highest response is judged to be the location of the object.

An expert systems approach makes use of knowledge of objects and their relationships in order to do a directed search. Knowledge of objects consists of measurements of their distinguishing

attributes. Care must be taken in representing geometrical information so that it is useful even if differences in scaling, translation, or rotation occur.

Relationships among objects in an image, such as included-in, left-of, or above, can also be described in the knowledge base. This information can help to direct the search for objects in the image. For example, if a control panel contains 3 rows of switches, and we are seeking a switch located on the second row, these relationships will direct the search.

A goal of this approach is to provide a generalized system which can be used in different environments for different object recognition tasks. The system should be expandable so that descriptions of new objects can be added to the knowledge base by users, and recognition of new objects will not require added programming.

The idea of using a knowledge base is to perform a focused, directed search of an image, to find the most prominent features first and use their locations to find other, less prominent features. Although our system does not incorporate a probabilistic approach, rather than saying absolutely that a certain feature has been found, a system can have a measure of merit for a feature: Feature A has been found with probability, or confidence level, 80%. That way, if further searching does

not prove consistent with the existence of Feature A, the confidence level can be reduced and the feature identified as something other than Feature A.

In order to perform this focused search, the knowledge base should provide information on different environments in which scenes are encountered. For example, if the environment is the interior of the space station lab module, then the objects to be located generally are arranged in rows and columns on a solid background. If the environment is space, the features would be areas of bright intensity against a dark background. Within a single application, there may be several environments possible, such as the outsides of panels in the space station, and the insides of specific cabinets. Different environments would indicate different algorithms applied to the problem.

Given a command to search for a particular object, there should be a sequence of operations to be performed stored in the knowledge base. This sequence may have alternatives, based on what features can be found, and with what confidence level.

b. A Strategy for Object Recognition

This section describes components in a general strategy for object recognition. Some of the steps could be accomplished in a variety of ways. This points out a need for an expert systems

approach, so that the appropriate algorithm can be selected for each step, based on characteristics of the scenes being processed.

The approach described here is based on the premise that we will deal with a number of separate scenes, which can be described in terms of the objects they contain, and the relationships among the objects. We allow for changes in scaling, illumination, and translation, and for small changes in rotation. The first step in the object recognition process is to determine which scene is being observed. This can be called "coarse discrimination."

Once the scene has been classified, the problem becomes one of finding the objects in the observed scene and matching them to those in the stored scene. First we must determine how the knowledge of the scene layout and of the objects is to be represented. Then, the procedure for matching can be determined. Knowledge representation and scene matching are also discussed in this section.

(i) Coarse Discrimination

Distinguishing one scene from another may be called coarse discrimination. This may be done by the use of global parameters of the image, such as overall brightness or color.

Another approach is to distinguish scenes based on features found in the scenes. In terms of distinguishing among broad classes of scenes, these features would need to be rather high-level objects, such as identifying cultivated fields in an aerial photo, or trees in a forest scene on the ground. If the scenes to be distinguished are several specific scenes, for example, different scenes within a room or views of control panels, the features used to discriminate scenes may be lower-level. For example, a view of a control panel may contain three long, distinct vertical lines. To distinguish this panel, we look for these lines, without trying to determine the objects in which the lines appear. Higher-level features may be identified in terms of a pattern of lower-level features. For example, a grid of lights can be used as a discriminating feature without identifying it as a grid of lights as such. It may present itself as a series of lines of very high and uniform texture. We may look for these lines, then, in order to find the grid of lights.

There needn't be a separate distinguishing feature for each scene, although for small numbers of scenes this may be the case. For large numbers of scenes, we may have S scenes and n features, $n < S$. A scene can be identified based on the absence or presence of the n features. This can be represented as a binary string of length n for each scene. In

the best case, n features could distinguish among 2^n scenes, if each possible binary string representing the presence or absence of features is used.

(ii) Scene Representation

One task in designing an expert vision system is to determine the means of representing knowledge of scenes and objects. Research has been done on image databases, for storage and retrieval of images. The idea is to allow some symbolic query to facilitate retrieval of a desired image. This same idea can be used in matching incoming images with the stored descriptions of scenes. In the original image database approach, on one end is a user with a symbolic request, on the other end is a file of images, and in the middle is an indexing scheme which makes use of the symbolic request and matches it with some symbolic representation of stored images. In our application, on one end is an incoming image, on the other end is a set of stored symbolic representations of images, and in the middle is a mechanism for converting our input image to a symbolic representation and matching it with a representation on file. In both cases, there is a matching of scene representations. In the expert vision system case, the problem is less structured, since we can have multiple views of the same objects or scenes, and we need descriptions which work for the different views.

The addition of new information to the knowledge base in our system is not an automatic process. Ideally, objects should be added to the knowledge base by showing the system pictures of the object at different orientations and having the system extract knowledge about the object and store it. It should check for similar objects and ask the user to provide differentiating features if the description is indistinguishable from another object. This part of an expert vision system is beyond the scope of our work.

In the space station application, the scenes are generally simple structured 2-D panels, with parts that remain nearly stationary. We have considered two approaches for representation of relationships among objects: 2-D strings and graphs.

2-D Strings

The first approach we examined for scene knowledge representation was the use of 2-D strings. The idea of a 2-D string was developed by S.K. Chang et. al.[11]. The strings are created by the use of enclosing rectangles within objects which are used as "points of view." From each point of view, there will be other objects which lie directly above, left of, right of, or below the object. These relationships are described in two strings, one for up-down relationships and the other for left-right relationships. This technique can be simplified by using

centroids of objects rather than enclosing rectangles. Instead of requiring an object to be directly above another in order to be considered "above," we may say that object A is above object B if the X pixel coordinate of A's centroid is significantly less than the X coordinate of B's centroid. This eliminates the need for splitting objects into several parts, as is done in method outlined in the paper.

The 2-D strings can be supplemented by another file that lists, for each object, the objects that enclose it. This can be used to direct the search for the object, by looking for the outermost enclosing object first, and working inward toward the desired object. Within each level of enclosure, the 2-D strings can be used to direct the search. We determine which edge (top, bottom, left, right) the desired object is closest to, in terms of how many other objects lie between it and the edge, and then work our way inward from the edge, finding successive objects in the 2-D string.

Other information is stored by the use of attributes for each object type listed in a string. For the space station, and other highly structured scene environments, objects can be classified as members of different object types. For example, SWITCH is an object type, and each individual switch is an instance of this type. BUTTON, RECTANGLE, and PANEL are other examples of object types for which many instances exist. For each object type, we need to define a

set of attributes which need not completely describe the object, but must help to distinguish it from other object types. Possibilities include relative area, circularity, rectangularity, and texture. The attributes used should be invariant to scale and rotation. It is hoped that the same set of attributes could be used to describe every object type that occurs on the space station panels. In some cases, we may want to use a "don't care" value, rather than an actual measure, such as in the case of the area of a rectangle. Rectangles on the space station panels usually enclose sequences of switches, but their size can vary widely.

The above-below and left-right strings are not useful in many different environments. Strings representing other relationships would be useful in other environments and could make use of the same principles of substring matching and string distance measurements. Such things as texture or intensity of objects within a given region could be ordered and expressed as a string. R1<R2<R3,R4 would indicate that the intensity of R1 is less than that of R2 which is less than that of R3 and R4 which are about the same intensity. Actually, areas, rectangularity, and circularity measures could be handled similarly rather than using an attribute file. The drawback is that we would no longer be taking advantage of the fact that objects of the same type have

similar measurements. There would need to be more data stored, since the relationship between each object at each inclusion level would be described.

Advantages of 2-D String Approach

2-D strings provide a simple, intuitive method of scene description. It is consistent, in that large, composite objects and small, primitive objects are all described in the same way. They each have a set of attribute measurements and two strings, one for above-below relations of constituent objects, and one for left-right relations. Primitive objects which contain no other objects will have null strings.

One problem in matching scenes is that not all objects may be completely visible in the image. With the use of 2-D strings, a representation of an incoming image can be matched against stored representations of scenes, and the "string distance" between the input and the stored scenes can be calculated. This is similar to the "string distance" measure used in spelling checkers, in which the strings are actually words. The scene which has the smallest distance from the input will be assumed to match our image.

Another problem is that the camera may be positioned in such a way that only a part of a composite object, such as a panel or subpanel, may be visible. Again, 2-D strings provide an advantage in dealing with this problem. The system may try to match an input image's string with substrings of strings which are on file to describe scenes. Alternatively, the strings stored in the knowledge base can be used as templates, for which we try to find appropriate objects in the incoming image to fill in the blanks.

In comparison with strict geometrical approaches which can be used for highly structured scenes such as ours, the 2-D string approach is more flexible in terms of scaling and rotation. A camera needn't be in exactly the same position each time a scene is viewed in order to be able to match the scene.

Disadvantages

The 2-D string approach is tailored for use in problems in which the scene's objects are stationary. It would not be useful in describing less structured scenes such as a desktop, on which objects can change positions, or in describing objects which can change shape, such as animals or jointed objects.

The string method is tolerant of small amounts of rotation, but if there is a large amount of rotation, the incoming scene must be normalized so that the strings describing the input and a stored string can be matched.

Graphical Approach

After working with the test images for our system, we came to realize that locating enclosing objects such as rectangles around rows of switches, can be a difficult task. Since these rectangles are not actually objects of interest, we decided to break each scene into a small number of subscenes, and represent all the relationships among the lowest-level objects (e.g. switches) within each subscene.

A general approach is to represent these subscenes as collections of graphs. Several relations among objects can be used to derive several graphs to describe a scene. Examples are left-of, above, included-by, adjacent-to. Matching of scenes consists of finding isomorphisms between the set of stored graphs and the set of observed graphs. There are methods for determining distances between two graphs as well as between two strings.

Graphs are superior to 2-D strings in representing complex relationships among objects, such as adjacency. Graph matching techniques may be more complicated, but algorithms

for graph matching have been widely studied.

This graph representation approach is also applicable to 3-D object recognition [12], and to character recognition [18]. The primitives in the scene that map to nodes in graphs can be lines, curves, or faces of polyhedral objects, rather than regions as in our application.

(iii) Scene Matching

The scene matching process is difficult because image segmentation seldom if ever produces results that match exactly with a stored representation of a scene. If a graphical matching approach is used, the first step is to represent an observed scene as a set of graphs. The presence of extra objects in the observed scene that are not present in the stored scene will make subgraph matching (in which the observed scene is a subgraph of the stored scene) ineffective. A sensible approach to the problem is to use scene and object knowledge to eliminate extra objects from the observed graphs before attempting to match.

This elimination can be carried out in several ways. One way is to first determine by some means probable object centers, and then match the objects present in the input to these object centers. For different types of images, different methods for finding probable centers of objects could be used. If an object's color is known, the image could be searched to

find a patch of that color as a starting point. Likewise, any other attribute of an object, such as intensity or texture could be used to find a patch from which to start.

Any object that is not positioned near a probable object center is not an object described in the knowledge base, and can be ignored. Another approach is to eliminate objects based on measurements of some of their attributes. In a given scene, if the scene consists of man-made objects, we can classify each object as belonging to a particular class of objects (e.g. switches, buttons, etc.). For each scene, we know what object classes will appear. Each object class is distinguished by a set of ranges for attribute values. Any object that does not measure as being within the ranges of any object class in that scene can be ignored in the matching process.

Another problem is that in observed scenes there will be missing or occluded objects. Matching through subgraph isomorphism is suitable for handling this partial matching of scenes. Once a matching of scenes is obtained for the objects that are observable, the location of missing objects can be postulated from the information in the stored graphs on relative object locations.

Due to noise, shadows, and limitations in the object finding algorithms, objects in observed scenes may have widely varying

attribute measures. The attributes used to distinguish among object types must be selected wisely. The introduction of fault-tolerance into this object classification has been studied [15][16][18]. Some researchers use a probabilistic approach, in which the stored nodes of graphs are considered as random variables, belonging to different classes with given probabilities. This approach does not seem appropriate for applications such as ours, in which a particular object is a member of a certain class of objects with probability one, and how that object might appear in an observed scene may be hard to determine. We can deal with this problem by allowing an observed object to be classified initially as a possible member of several object classes. Based on attribute measures, some object classes may be immediately ruled out, but others may be possibilities to consider. This is taken into account in the graph matching procedure, limiting the possible mappings of observed to stored nodes.

Another problem is that changes in scaling, rotation, or illumination may alter the relations among observed objects. For example, different illumination angles onto shiny objects can cause relative intensities of the objects to be different. This problem must be considered when choosing which relations to use in describing the scenes. If a standard subgraph matching algorithm is used, missing arcs cause no problems, but extra arcs in an observed graph will make it impossible to find a subgraph isomorphism. Some researchers use a

probabilistic approach in which an arc is considered a random variable taking on different values with different probabilities [18]. This does help to deal with the problem, but also makes the scene matching process more complicated. Many isomorphisms will be found, which must be evaluated to determine which is the most likely match.

With relations such as left-of and above, significant rotation would be an insurmountable problem if the scene could not be rotation normalized. In some types of scenes, normalization may be achieved by locating a prominent feature and determining its orientation.

In scenes in which only slight rotation is expected, we can deal with this problem by using overlapping ranges. In other words, an object may be left-of another, above it, or both, if it falls within the overlapping range. In graphical terms, this translates into having an extra arc in the stored graph representation of the scene. Then, in the observed scene, an object in the overlap region is determined to be either left-of or above the other object. There will be one missing arc in the observed graph, which is no problem in subgraph isomorphism algorithms.

c. The Demonstration System

A large part of the work on this contract has been devoted to the development of a demonstration machine vision system, implementing the basic task of object recognition.

We have used a Perceptics 9200e Image Processor, on a host VAXstation, to develop the system. The Perceptics is a general purpose image processing system, which implements many lower-level routines that are necessary in any machine vision system. Other routines have been developed in C on the host machine, which can access the image memory of the Perceptics.

In order to use realistic test images, we have used photographs of actual space shuttle simulator panels. One of these is shown in Figure 2.

The following sections describe the major steps in the processing of the system. These steps are shown in Figure 3. Also included is a description of the library of routines used, some developed on the VAXstation, others provided with the Perceptics.

(i) Scene Identification

The first step in the demonstration system is to identify which scene is present in the input image. To identify the scene, the system searches for primitive distinguishing features. The presence of absence of the features in the input image is matched with lists of features present for each

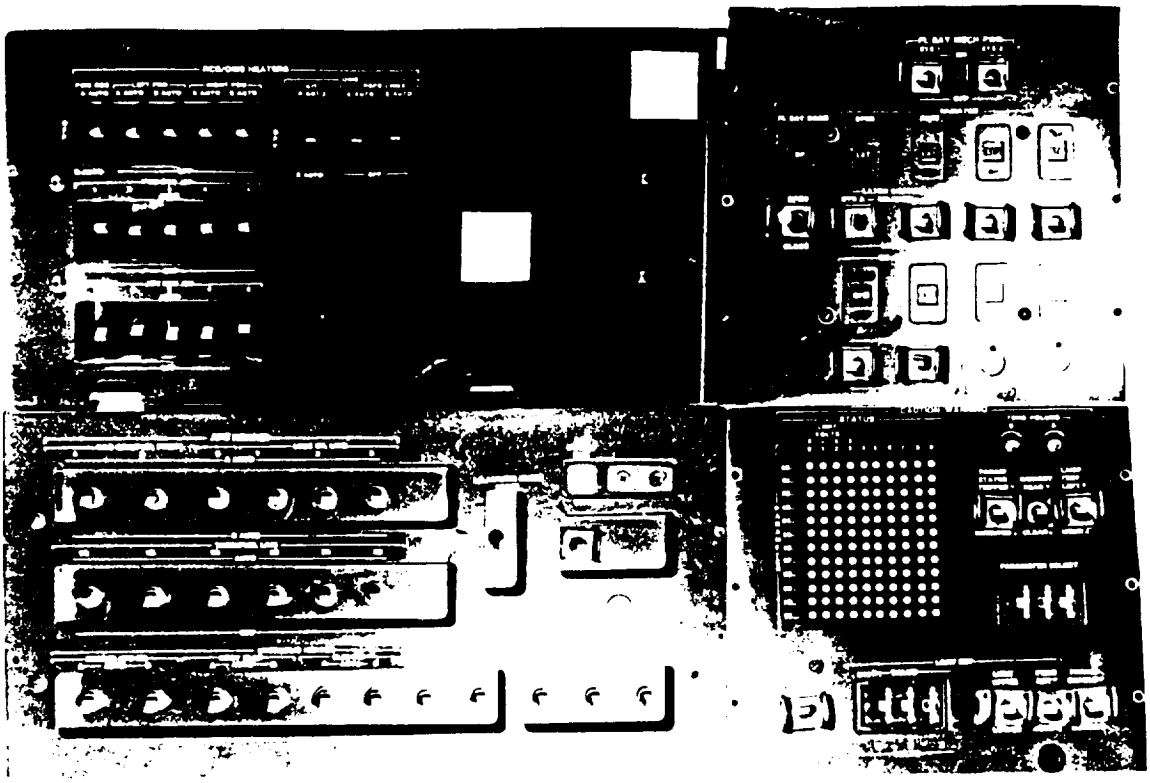

Figure 2: One of the scenes used as a realistic test image for the system.

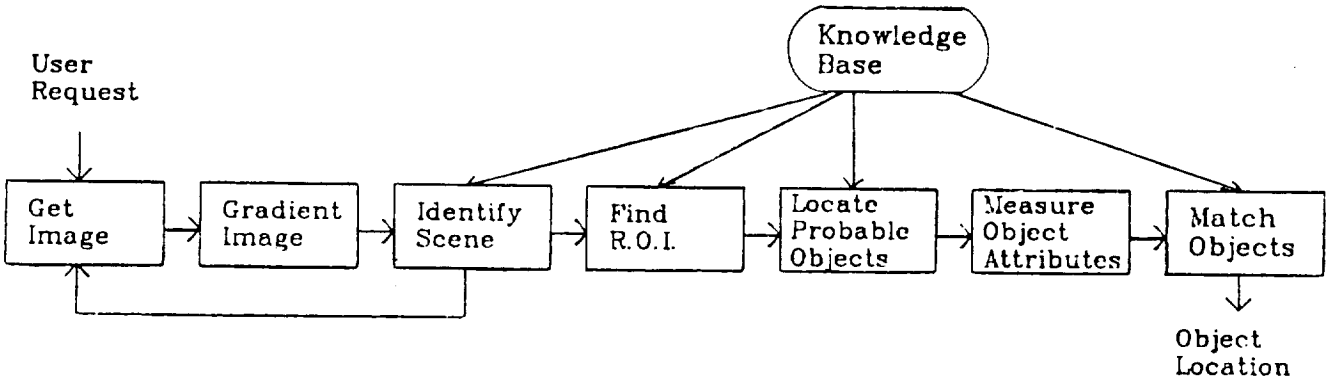

Figure 3: The steps in the object recognition process.

scene in the knowledge base. Presently, a scene must have features that match exactly with one of the scenes in the knowledge base. It would be possible to allow for closest matches by computing the string distance between a binary string denoting presence and absence of features in the input image with the strings in the knowledge base, and assume the scene to be the one with the closest match.

The features used for scene identification are sets of lines in the gradient image with certain characteristics. These lines correspond to edges in the original image.

Figure 4 depicts the scene identification process for the panel of Figure 2. In this example, lines of high texture, as measured by high average difference between neighboring pixels, are found through the columns of an array of lights, and also through a row of switches. The diagonal line and the set of lines in the upper left corner of the image represent the best matches for two additional features that are present on other panels but not on this panel.

We use primitive features to keep processing for scene identification to a minimum, but any features could be used, as long as the process for finding them could be listed in the knowledge base.

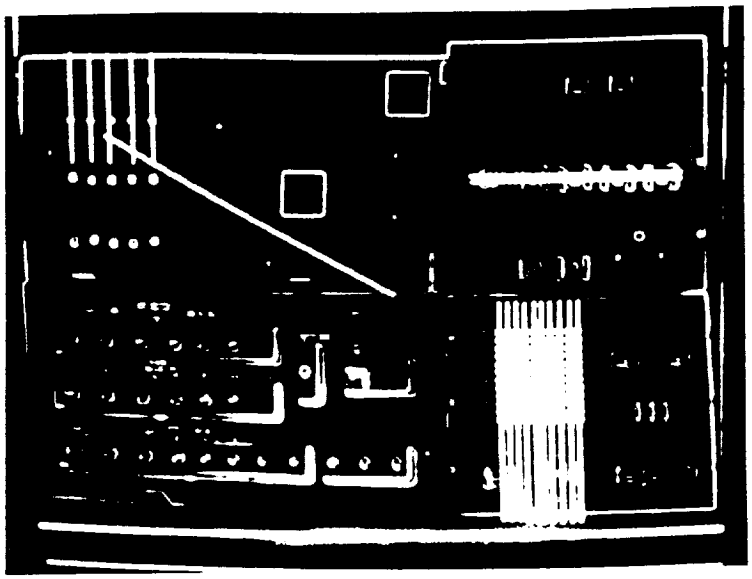

Figure 4: The scene identification process performed on the gradient image of the panel in Figure 2.

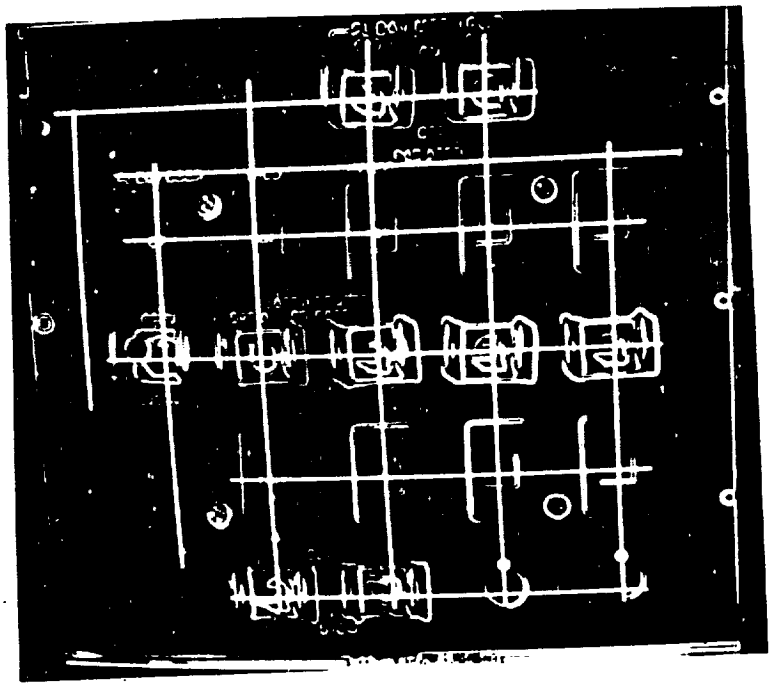

Figure 5: Determination of likely starting points for objects, performed on a sub-panel of Figure 2.

(ii) Locations of Objects in Observed Scene

Given the location of features in an input image, it is possible to compute coordinates for a region of interest of the scene that contains the desired object.

Once an image of the region of interest of the scene is obtained, we find starting points of probable objects. In scenes consisting of well-separated blobs on a background, a method that has proven useful is to search for a specified number of horizontal and vertical lines of high texture, with some minimum spacing between them. Figure 5 shows the result of this process on one of our regions of interest. Most of the intersection points pass through objects on the image. There are some false lines, since there is some printing on the control panels that results in high-texture lines.

The minimum spacing chosen is large enough to prevent the appearance of more than one line in the same direction through the same objects. Only a minimum is given so that the object seed points may be found for images that are translated or scaled differently.

The intersection points of lines found define possible object locations. We then perform a blob-finding routine that was provided with the Perceptics software. The routine finds "objects" in the scene that fall within a specified area range. Naturally, the set of "objects" found includes many

that are not defined in the knowledge base. Shadows, marks and printing on the control panels, and other regions that do not correspond to objects will be found by the blob-finding routine. In order to eliminate these non-objects from consideration, the possible object locations found in the previous step are used. Blobs whose centroids are significantly distant from all intersection points will be assumed not to correspond to objects described in the knowledge base.

(iii) Representation of Input Scene as a Grid of Objects

The control panels contain instances of a finite number of object types, e.g. switches, buttons, knobs, etc. For each object type, the knowledge base contains an acceptable range of values for each attribute. The attributes used may differ depending on the object type. For example, circularity may be a good attribute to use for knobs, but texture may be better for switches enclosed in brackets. Since the possible objects in the input image may be processed in parallel, it may be worthwhile to measure all attributes, even though some results may not be used. Once the attribute measures have been determined, the knowledge base is consulted to determine for each possible object, the set of object types consistent with its measurements. For example, Object 1 may "look like" a

switch or a button. Some possible objects will not match to any object types. These will be eliminated from further processing.

For simplicity, the demonstration system represents scenes as grids, i.e. rows and columns of objects. This provides a simple way of denoting above-below and left-right relations among objects. Several grids could be used to describe one scene. For each object, the knowledge base indicates to which grid the object belongs.

A more general approach is to assign each object in the observed scene a number, and create a set of graphs describing their relationships. A vertex mapping matrix would then be created, which shows possible mappings between the observed objects and the stored ones. These mappings are determined based on degree of the vertices and the types of objects to which they may correspond.

(iv) Matching of Observed Objects with Stored Objects

The next step is to match the stored grid of objects with the observed grid. We eliminate rows or columns of the observed grid that do not contain actual objects. These usually occur because of vertical or horizontal lines of high texture that go through areas of printing on the control panels. Then, the observed grid and the stored grid should match. We calculate the difference between observed and stored, i.e. how many

missing or extra objects occur. If the difference is acceptable, we assume a match with the stored grid. The position of the desired object on the Perceptics screen is shown.

In a more general approach, the system would use a subgraph isomorphism algorithm to find a matching of observed to stored objects.

(v) Library of Routines

In any image processing system, there must be a library of routines to do basic functions such as finding lines or shapes in the image. In this section we describe some of the major routines used in the system. Some are built into the Perceptics and some were programmed on the VAX. Such routines as finding the gradient image, thresholding, histogramming, finding blobs and measuring some of their attributes, are provided in the Perceptics software. A set of line finding routines was programmed on the VAX.

Perceptics Blob Finder

The blob-finding routine built into the Perceptics provides very useful results. It finds objects within a given size range in the image, and computes a number of useful statistics for each object. The blob finder can be run interactively or called from within a C program. The

demonstration program calls the blob finder and stores its results in a file on the VAX that is later used by the program.

The attributes measured for the blobs include area, perimeter, width, height, and circularity. Many additional statistics are optional. Unfortunately, we have found that the statistics that are based on the perimeter measure are often unreliable. One problem arises if the boundary of an object is not a solid line. If the boundary is broken, the perimeter outlines the outside and inside of the boundary, giving a number roughly double what is expected. Statistics based on width and height are much less sensitive to these sorts of problems.

These attributes are currently used to determine if a blob is a valid object in the scene, or is not valid and should be ignored.

Line Finders

One prominent feature in most images is the set of lines, usually from an edge image. If we know that a scene should have a certain set of lines, these can be searched for in particular windows, spacings, and orientation ranges. A window can eliminate looking in areas of the scene where a line is not likely. Specifying a minimum spacing between

lines eliminates some duplication, such as finding two lines which are really internal to one thick line. Orientation ranges also help narrow down the extent of the search. If an image is supposed to have a line in a certain window oriented at about 10 degrees, the search can be restricted to lines with orientations of within 5 to 15 degrees.

A line can be detected at a given orientation by finding the line of pixels in that direction with the maximum intensity sum. If we use this information alone, there may be a problem. Consider the case where along one row in the image, there are 50 white pixels which are scattered along the line and do not appear to constitute a single bright line. Another row in the image may contain 30 white pixels which comprise a single solid line segment. If we decide on the basis of intensity sum alone, the first row is considered a better line. So, finding endpoints of a line and characterizing the brightness of the line between the endpoints can give a better measure of the strength of a line. In order to specify endpoints, we could start by assuming that the first and last white pixel along the line are the endpoints, then see what percentage of that segment are white pixels.

Another measure which is useful in finding lines is the sum of absolute differences of successive pixels along a prospective line. In a "good" line, this sum is small: the

line is relatively uniform. In a "poor" line, the sum is large: there may be many places along the "line" which change abruptly from one intensity value to a very different value. This could happen in the case of having several scattered very bright points along a line, which are not truly connected and do not define a line. A linear combination of the measure of intensity difference between the current line and the line of pixels above and below it, and the measure of absolute difference of successive pixels along the line, can be used to measure the goodness of a line. The intensity difference between lines should be high, and the difference along the line should be low.

The line finders developed for the demonstration system incorporate the ideas discussed above. In addition to vertical and horizontal line finders, there is an angle line finder that searches for lines that fall within a given range of angles. All line finders search within a specified window of the image. The average intensity of a pixel along every line in the window is computed, along with the average difference among neighboring pixels. Then, this array of lines can be searched for a specified number of "best" lines (based on high intensity and low difference), brightest lines, or most textured (highest difference) lines. All of these criteria for lines are useful in some situations.

IV. Conclusions and Recommendations

The implementation of the demonstration system on the Perceptics has provided useful experience in the use of the knowledge based approach to vision systems. The demonstration system now has some important capabilities, and can find many objects in the sample scenes. More work is needed to enhance these capabilities and to structure knowledge about object locations in a better way.

Research into scene representation and matching has been fruitful, and may lead to further development of innovative graph-based approaches in the contract continuation. In addition, research into neural networks has laid a groundwork for future integration of this approach into the demonstration system, and for evaluation of neural networks in vision problems in general.

One conclusion that has become apparent in the course of our work is that no one approach to machine vision will be applicable to all problems. The need for expert vision systems, to choose appropriate procedures for different situations, is evident.

In the continuation contract, the following tasks are to be performed:

1) To complete and demonstrate the knowledge based system which is being developed on the Perceptics image processing system.

2) To evaluate the previously developed knowledge base and the overall system performance for real time operation.

3) To provide preliminary design for hardware implementation, and to evaluate the commercially available high speed signal processing chips for utilization in this design.

4) To determine the feasibility of incorporating neural network technology (back propagation technique and harmony model approach) into design concept.

5) To investigate the possibility of combining expert vision system with neural network technology to form a hybrid vision system which inherits the merits of both technologies.

REFERENCES

1. Frank L. Vinz, Linda L. Brewster, and Dale Thomas, " Computer vision for real-time orbital operation ", NASA technical report TM 86457, Marshall Space Flight Center, Huntsville, AL, August 1984.

2. Ballard, D. H. and Brown, C. M.," Computer Vision ", Prentice Hall, Englewood Cliffs, New Jersey, 1982.

3. S. L. Tanimoto, "A comparison of some image processing methods", IEEE conference on Pattern Recognition and Image Processing, Chicago, IL., pp. 280-286, May 31-June 2, 1978.

4. Vanderbrug and Rosenfeld, "Two stage template matching", IEEE Trans. on Computers, Vol. C-26, pp. 384-393, April 1977.

5. Dudani, Jenney and Bullock, "Correlation and alternatives for scene matching", IEEE conference on Decision and Control, Clearwater, FL., pp. 774-779, Dec. 1-3, 1976.

6. M. K. HU, "Visual pattern recognition by moments invariants", IRE Trans., It-8, pp. 179-187, February 1962.

7. Cohen, P., and Feigenbaum, E.," The handbook of artificial intelligence ", Volume III, William Kaufmann, Los Altos, CA., 1982.

8. Gonzalez and Wintz, "Digital image processing", Addison-Wesley publishing company, Inc., 1977.

9. Novak, "Correlation algorithm for radar map matching", IEEE conference on Decision and Control, Clearwater, FL, pp. 780-790, Dec. 1-3, 1976.

10.Rosenfeld, "Expert vision systems: Some issues," Computer Vision, Graphics, and Image Processing, pp. 99-117, 1986.

11. S.K. Chang et.al., "An intelligent image database system," IEEE Transactions on Software Engineering, May 1988, pp. 681-688.

12. J.R. Englebrecht, F.M. Wahl, "Polyhedral object recognition using Hough-space features," Pattern Recognition, Vol. 21, No. 2, 1988, pp. 155-167.

13. R. Greene, "Scene knowledge representation for expert vision systems," Ph.D. dissertation, University of Alabama in Huntsville, 1988.

14. L.G. Shapiro, R.M. Haralick, "A metric for comparing relational descriptions," IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. PAMI-7, No. 1, 1985, pp. 90-94.

15. L.G. Shapiro, R.M. Haralick, "Structural descriptions and inexact matching," IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. PAMI-3, No. 5, Sept. 1981, pp. 504-519.

16. W. Tsai and K. Fu, "Subgraph error-correcting isomorphisms for syntactic pattern recognition," IEEE Transactions on Systems, Man, and Cybernetics, Vol. SMC-13, No. 1, January/February 1983, pp. 48-62.

17. J.R. Ullmann, "An algorithm for subgraph isomorphism," Journal of the Association for Computing Machinery, Vol. 23, No. 1, January 1976, pp. 31-42.

18. A.K.C. Wong, M. You, "Entropy and distance measures of random graphs," Proceedings of the Conference on Computer Vision and Pattern Recognition, IEEE, 1983, pp. 371-376.